Journal of Information Security and Applications 60 (2021) 102865

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Event-oriented linkable and traceable anonymous authentication and its application to voting

Peng Li, Junzuo Lai *, Yongdong Wu

College of Information Science and Technology, Jinan University, Guangzhou 510632, China

ARTICLE INFO

Keywords:
Anonymous authentication
Linkability
Traceability
Voting
Blockchain

ABSTRACT

In the pursuit of anonymous authentication schemes that balance anonymity and accountability, various authentication schemes have been proposed. However, most of existing schemes cannot achieve linkability while holding public traceability. In this paper, we introduce a new variant of anonymous authentication called *event-oriented linkable and traceable anonymous authentication* (EOLTAA) and provide a generic construction. In an EOLTAA scheme, a message to be authenticated binds an *event*. If two different messages binding the same event are authenticated by an identical user, anyone can link the two authentications and further reveal the user's identity. Then we formally define the security requirements of EOLTAA, including unforgeability, anonymity, linkability and public traceability. We give a generic construction satisfying the security requirements. With this new authentication scheme, we construct the first decentralized, anonymous, linkable and publicly traceable e-voting based on blockchain.

1. Introduction

In the age of information technology, individuals electronically participate in activities more than ever, so that people can share information, express opinions, purchase goods, book flights and hotels over the Internet [1]. Meanwhile, people are more focus on the protection of privacy. For example, we may not want others to pay attention to what you said, what you bought and where you will go. It is therefore paramount to take actions to protect the privacy of users in the applications.

Anonymity is a significant form of privacy protection. On the other hand, if an application only ensures the anonymity, the recipient may cannot be convinced of the integrity of transmitted messages. Since malicious adversary may modify the messages that will be transferred, or they merely want to transmit false messages without being identified. Thus, it is critical to ensure the validity of a message in the anonymous communication. One solution is to require the user to anonymously authenticate the delivered messages.

Anonymous authentication plays a critical role in authenticating users while maintaining their privacy. In e-voting, the verifier should be able to verify the validity of a ballot without knowing the real identity of the corresponding voter. What's more, many applications, such as e-cash, e-auction, e-voting, usually enable participants to anonymously take part in, but anonymity in turn leads to dishonest behaviors, such as double spending in e-cash and double submission in e-voting. In particular, these one-round systems always require users to involve in only once, but misbehaving users may submit twice or more to abuse this anonymity to disturb the rules. For example, each voter is not allowed to vote twice in a voting numbered *event*, but he is able to join in other votings with different serial numbers. Thus, measures should be taken to restrict the behavior of these malicious users who attempt to engage in multiple times.

Accountable anonymous authentication can further realize the trade-off between anonymity and accountability through detecting malicious behaviors. For example, if a voter votes twice in an e-voting with serial number *event*, then the double voting can be detected, and maybe the user's identity can also be traced. Existing schemes, such as direct anonymous attestation with linkability [2], linkable group signatures [3,4], linkable ring signatures [5], linkable anonymous credentials [6], accountable ring signatures [7] and traceable attribute-based signatures [8,9], have been proposed to satisfy various degrees of accountability. For example, some linkable mechanisms only aim to detect whether two signatures are from the same signer without traceability, and some traceable mechanisms prefer to find the identity of a malicious user.

Motivation. To achieve accountability, numerous linkable mechanisms and traceable mechanisms are designed to fulfill different requirements. Compared with the tracing performed by trusted parities, it is desirable to realize the public tracing mechanism that enables anyone to perform

* Corresponding author.
E-mail addresses: penglijnu@gmail.com (P. Li), laijunzuo@gmail.com (J. Lai), wuyd007@qq.com (Y. Wu).

https://doi.org/10.1016/j.jisa.2021.102865

Available online 19 May 2021

Table 1
Comparison between our scheme and related schemes.

Scheme	Anonymity	Linkability	Traceability	Public traceability	Generic construction
[10–12]	✓	×	×	×	✓
[2,5,13]	✓	✓	×	×	×
[14,15]	✓	✓	×	×	✓
[9,16–18]	✓	×	✓	×	×
[7,8,19]	✓	×	✓	×	✓
[6,20]	✓	✓	✓	×	×
[3,4,21]	✓	✓	✓	×	✓
[22–26]	✓	✓	✓	✓	×
Ours	✓	✓	✓	✓	✓

and cannot compromise others privacy if someone misbehaves. However, current public tracing mechanisms in anonymous authentications have a major limitation: there is no generic construction to achieve public linking and public tracing so as to be compatible with different basic building blocks. We are motivated to put forward an anonymous authentication with a generic construction to achieve public linking and public tracing, and make the anonymous authentication to be more easily obtained.

Our Contribution. In this paper, we concentrate on a new variant of anonymous authentication scheme. We achieve the event-oriented linkability and traceability in which anyone can determine whether two authentications are linked and can further trace a user, if and only if two different messages with an identical event are authenticated by the same user. Specifically,

1. We present a new variant of anonymous authentication called *event-oriented linkable and traceable anonymous authentication* (EOLTAA) and provide a generic construction. We formally give the security requirements of EOLTAA, including unforgeability, anonymity, linkability and traceability. A message to be authenticated binds an *event* in an EOLTAA scheme that holds (i) unforgeability, i.e., anyone cannot forge an authentication in the name of a certain user; (ii) event-oriented linkability, if two different messages binding a common event are authenticated by the same user; (iii) public traceability, i.e., anyone can reveal a user's identity if the user authenticates two different messages with the same event; (iv) anonymity and unlinkability, as long as a user authenticates either a message only once, or various different messages that own distinct events.
2. We give a generic construction of EOLTAA to satisfy the above mentioned security requirements, and reduce the security to a set of assumptions. We also provide a comparison between our scheme and other works on anonymous authentication schemes in Table 1. If lattice-based signature schemes and zk-SNARK protocols are exploited, we can obtain the first lattice-based anonymous authentication scheme with linkability and traceability so as to resist quantum attacks.
3. We further utilize the new authentication scheme to construct the first decentralized, anonymous, linkable and publicly traceable voting scheme based on blockchain, and aim to address the double voting without any help from other parties.

The remainder of the paper is organized as follows. In Section 2, we review some related work. In Section 3, we give some preliminaries. Then we define our event-oriented linkable and traceable anonymous authentication in Section 4. Section 5 contains the construction and security theorems. We further propose a voting protocol based on blockchain in Section 6, and conclude in Section 7.

2. Related work

2.1. Anonymous authentication

Anonymous authentication offers a great privacy protection to users. Direct anonymous attestation [27] is the method for remote authentication of a hardware module while preserving the privacy of the user of the platform that contains the module. It can be considered as a group signature scheme without the feature that a signature can be opened. [28] is an efficient direct anonymous attestation scheme such that it is suitable for resource constrained environments. An anonymous signature [29], as a special kind of digital signature scheme, is designed to ensure that signatures do not reveal the signer's identity. Based on [29], [30] further presents a general and efficient way to construct such anonymous signature schemes. However, they only ensure the basic anonymity of users without linking or tracing.

Group-oriented signature schemes play an important role in anonymous authentication. In a ring signature scheme [31], any member of a group can anonymously sign messages on behalf of the group, and it is impossible to decide whether two signatures are issued by the same group member [32], i.e., there is no linkability. Some practical ring signature schemes and their variants have been present, such as identity-based ring signature [33], attribute-based ring signature [34], threshold ring signature [35]. These schemes achieve anonymity without accountability.

Some anonymous authentication schemes are designed to combine with an access policy. Policy-based signatures [36] ensure that a signature does not reveal the policy associated to the signing key and neither the witness that is used to create the signature. An attribute-based signature [12] allows a user, whose attributes satisfy a signing policy, to sign a message without revealing any other information about the signer's identity or attributes. Decentralized attribute-based signatures [37] are designed to remove the central authority and the trusted setup in the multi-authority attribute-based signatures. Anonymous credentials [38] enable the authentication of users without revealing their identity. In an anonymous credential system [11,39], users authenticate themselves by attesting the possession of credentials while not disclosing any other information about themselves [40]. Usually, it is unlinkable to previous uses of the same credential, or to any other identifying information about the user. Some works further extended the anonymous credential schemes to support delegation [41], revocation [18], update [10], hidden policies [42], blacklist [43] and so on. All the above mentioned schemes cannot realize linking and tracing.

2.2. Accountable anonymous authentication

Accountable anonymous authentication can achieve the possibility to detect double authentication, or to directly reveal the misbehaved user's identity, instead of ignoring the fact that users may misbehave without worrying about being punished. Linkable group signature [6, 44] and linkable ring signature [5,14] have the common functionality of determining whether two given signatures are made by the same group member. Similarly, [2] is an anonymous attestation scheme that also achieves the same linkability. Attribute-based signatures with controllable linkability can check if two signatures are created by the same signer without breaking anonymity, and [13] allows an entity who has a linking key to perform the detection, but [15] achieves the public linking that can be run by any party. Linkable anonymous authentication [45] enables everyone to link the authentications if the same key holder authenticates two messages sharing the same prefix. We can consider this kind of linkability as a lightweight accountability, which cannot further achieve traceability.

A stronger accountability is just like to trace the user and reveal the corresponding identity, so as to avoid abuse of anonymity. In a group signature scheme [46], any group member can anonymously sign messages on behalf of the group. However, to settle a dispute, any signer's

identity can be traced by a group manager. In [19], authors designed a fully-anonymous group signature to implement privacy-friendly authentication mechanism, and there is an opening authority to track a user's identity. Additional, the traceable designated verifier signatures in [17], accountable ring signatures in [7], traceable attribute-based signatures in [8,9] and attribute-based anonymous credentials in [16] introduce a trust third party to reveal the user's identity, but these schemes do not implement the above linkability.

Some schemes achieve the linkability and traceability simultaneously with the help of trusted parities. In [47] and [20], a linking authority can link signatures without identifying the signers, and an opener can open a signature to ensure the traceability. In [4], the auditing manager is introduced to determine whether two signatures come from the same signer, and the supervision manager has the ability to further trace the signer's identity. In [21], the link manager and trace manager implement the linking and tracing, respectively. In [3], an opener can show if a signature is created by a specific member, and can prove whether two signatures originate from the same signer without disclosing anything else. In [48], the authors introduces a revocable and linkable ring signature scheme, in which two signatures created by the same user can be linked and a revocation authority is able to revoke the anonymity of a certain user. All these above schemes are not publicly traceable.

2.3. Publicly traceable anonymous authentication

[22] is a separable linkable threshold ring signature scheme, which allows anyone to determine if two signatures are created from the same group member and identify the member's identity. [26] is a tracing-by-linking group signature scheme, which enables a public algorithm to trace a member's identity without needing any trapdoor if he signs twice. [49] is a k-times anonymous authentication scheme in which a user can be anonymously authenticated as long as the number that he is authenticated is within an allowable number. If a user is authenticated beyond the allowable number, anyone can trace him without help from the authority. [24] is an event-oriented k-times revocable-iff-linked group signature scheme, which ensures that no party can revoke the anonymity of a signer, but everyone can identify the signer if he signs more than k times for a particular event. [50] is a revocable-iff-linked ring signature scheme, which makes a signer to be linked and his identity to be revoked by everyone if he signs twice or more. [51] and [25] introduce a traceable ring signature scheme, in which anyone who creates two signatures for different messages with respect to the same tag can be publicly traced. [23] proposes an ID-based linkable and revocable-iff-linked ring signature such that everyone can compute a signer's identity if he creates two linkable ring signatures in the same event. These schemes ensure the property of public traceability, but they do not have a generic construction.

3. Preliminaries

3.1. Digital signatures

Digital signatures can be used to authenticate the origin and the integrity of a message [52]. A digital signature scheme consists of three algorithm: (Setup, Sign, Verify).

- $\mathsf{Setup}(1^\lambda) \longrightarrow (pk, sk)$ is a algorithm which takes as input a security parameter λ and outputs a public key pk and a secret key sk.
- $\mathsf{Sign}(m, sk) \longrightarrow \sigma$ is a algorithm which takes as input a message m and a secret key sk, and produces a signature σ.
- $\mathsf{Verify}(m, pk, \sigma) \longrightarrow 0/1$ is a algorithm which takes as input a message m, a public key pk and a signature σ, and returns 1 or 0 for valid or invalid.

A digital signature is said to be secure if it satisfies existentially unforgeable under a chosen-message attack [53]. A signature scheme is existentially unforgeable if, given any polynomial number of pairs $(m_1, \sigma_1), (m_2, \sigma_2), \ldots, (m_k, \sigma_k)$, where σ denotes the signature on the message m, it is computationally infeasible to generate a pair (m_{k+1}, σ_{k+1}) for any message $m_{k+1} \notin \{m_1, \ldots, m_k\}$ [54]. This means that no efficient adversary who is given a signature for some messages of his choice can generate a signature for a new message with non-negligible probability.

Existential unforgeability against adaptive chosen message attacks for a signature scheme is defined using the following game.

Setup. The challenger runs Setup. It gives the adversary $\mathcal{A}$ the public key pk and keeps the secret key sk to itself.

Query. $\mathcal{A}$ issues signature queries $m_1, \ldots, m_q$. To each query m_i, $\mathcal{A}$ responds by running Sign to create a signature σ_i on m_i and sending σ_i to $\mathcal{A}$. These queries may be asked in an adaptive manner so that each query m_i may be dependent on the replies to $m_1, \ldots, m_{i-1}$.

Forge. $\mathcal{A}$ outputs a forgery (m^*, σ^*). $\mathcal{A}$ wins if σ^* is a valid signature of m^* and m^* is not among the messages m_i queried during the query phase.

Definition 1. A digital signature scheme is existentially unforgeable against adaptive chosen message attacks, if no probabilistic polynomial-time adversary $\mathcal{A}$ can win in above game with non-negligible probability.

3.2. zk-SNARKs

A zero-knowledge proof enables one to build a cryptographic proof so as to convince others that he indeed knows a secret witness without disclosing any additional information [55]. The tool is useful to prove that one correctly follows the protocol instead of deviating from it. For instance, for a public value x, one knows a certain witness w so that $f(w) = x$, where f is a public function.

The zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) [56] enables a proof to be generated non-interactively, and what's more, the proof is *succinct*, i.e., the proof is very short and easy to verify [57]. A zk-SNARK can be exploited for further achieving a kind of designated proof, just like to perform a protocol in order to generate a witness w such that $C(x, w) = 1$ with the circuit C and a statement x [58]. For a circuit $C : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$, its satisfaction problem is a relation $\mathcal{R}_c = \{(x, w) \in \{0,1\}^n \times \{0,1\}^h | C(x, w) = 1\}$, and its NP-complete language is $\mathcal{L}_c = \{x \in \{0,1\}^n | \exists w \in \{0,1\}^h, s.t., C(x, w) = 1\}$. Then, with a public x, one can attest a secret witness w satisfies $C(x, w) = 1$ such that $x \in \mathcal{L}_c$.

Specifically, a zk-SNARK is a triple of algorithms (Setup, Prover, Verifier).

- $\mathsf{Setup}(1^\lambda, C) \longrightarrow pp$ is a algorithm which, on input a security parameter λ and a circuit C, outputs the system's public parameters pp. The algorithm is only required to run once per circuit.
- $\mathsf{Prover}(pp, x, w) \longrightarrow \pi$ is a algorithm which, on input the public parameters pp, a public x and a witness w, outputs a proof π.
- $\mathsf{Verifier}(pp, x, \pi) \longrightarrow 0/1$ is a algorithm which, on input the public parameters pp, a public x and a proof π, outputs 1 if π is a valid proof such that $x \in \mathcal{L}_c$.

Therefore, one can exploit the above algorithms to establish a non-interactive and succinct proof for the language $\mathcal{L}_c$, and others can verify whether the proof is valid to attest a statement $x \in \mathcal{L}_c$.

The zk-SNARKs satisfy the following properties: (i) *Completeness*. Prover can generate a proof and convince the verifier that the prover knows a witness corresponding to the statement. (ii) *Soundness*. No polynomial-time adversary can forge a proof for a statement without

knowing its corresponding witness. (iii) *Proof of Knowledge*. There exists an extractor algorithm such that if a prover convinces a verifier, then given oracle access to the prover, the extractor algorithm can actually output the witness. (iv) *Zero-Knowledge*. The proof leaks nothing except for the statement. (v) *Succinctess*. The proof size is linear in the security parameter.

4. Event-oriented linkable and traceable anonymous authentication

4.1. Syntax

An event-oriented linkable and traceable anonymous authentication (EOLTAA) consists of the following seven algorithms.

- $\mathsf{CSetup}(1^\lambda) \longrightarrow (\mathsf{MPK}, \mathsf{MSK})$. The master key generation algorithm is a function that takes as input a security parameter λ, and outputs a master public key MPK and a master secret key MSK.
- $\mathsf{UKeyGen}(1^\lambda) \longrightarrow (upk, usk)$. The user key generation algorithm is a function that takes as input a security parameter λ, and outputs a public key upk and a secret key usk.
- $\mathsf{CertGen}(upk, \mathsf{MSK}) \longrightarrow \sigma$. The certificate generation algorithm takes as input a user's public key upk and the master secret key MSK. It outputs a certificate σ that can validate the corresponding public key upk.
- $\mathsf{Auth}(m = e \parallel p, upk, usk, \sigma, \mathsf{MPK}) \longrightarrow \pi$. The authentication algorithm takes as input a message m that has an event identifier e and a payload p, a user's public key upk, secret key usk, certificate σ, and the master public key MPK. It outputs an authentication token π on the message m.
- $\mathsf{Verify}(m, \pi, \mathsf{MPK}) \longrightarrow 0/1$. The verification algorithm takes as input a message m, an authentication token π and the master public key MPK. It outputs 1 or 0 to decide whether the proof is valid or not.
- $\mathsf{Link}(m_1, m_2, \pi_1, \pi_2) \longrightarrow 0/1$. The link algorithm takes as input two valid message and authentication token pairs (m_1, π_1), (m_2, π_2). It outputs 1 if two messages binding a common event are authenticated by the same user; otherwise, it outputs 0.
- $\mathsf{Trace}(m_1, m_2, \pi_1, \pi_2) \longrightarrow \perp / upk$. The trace algorithm takes as input two valid message and authentication token pairs (m_1, π_1), (m_2, π_2). It outputs a public key upk which points to the user who authenticates two messages binding a common event; otherwise, it outputs the error symbol $\perp$.

Any construction of EOLTAA must be correct.

Definition 2 (*Correctness*)**.** An EOLTAA scheme is correct if it has verification correctness, linking correctness and tracing correctness:

- *Verification correctness*. An authentication token produced by an honest user should be accepted by the Verify algorithm.
- *Linking correctness*. If two honestly created authentication tokens are determined as linked, then they must have been generated by an identical user with respect to the same event.
- *Tracing correctness*. In the case of linked, an honest user's public key will be revealed by the Trace algorithm.

4.2. Notions of security

Security of EOLTAA schemes has four aspects: unforgeability, anonymity, linkability and traceability.

Unforgeability. Unforgeability for EOLTAA schemes is defined as the following game between the Challenger C and the Adversary $\mathcal{A}$.

1. C runs $\mathsf{CSetup}(1^\lambda)$ to create a master key pair (MPK, MSK), and runs $\mathsf{UKeyGen}(1^\lambda)$ to generate n public-secret key pairs (upk_1, usk_1), $\dots$, (upk_n, usk_n), and gives MPK, $upk_1, \dots, upk_n$ to $\mathcal{A}$.
2. $\mathcal{A}$ gives C a public key $upk_i \in \{upk_1, \dots, upk_n\}$. C runs $\mathsf{CertGen}(upk_i, \mathsf{MSK})$ and returns a certificate σ_i.
3. $\mathcal{A}$ chooses and sends a public key upk_j and a message m_i to C, and asks C to authenticate m_i. Then, C runs $\mathsf{Auth}(m_i, upk_j, usk_j, \sigma_j, \mathsf{MPK})$ and outputs a corresponding authentication token π_i.
4. $\mathcal{A}$ selects a public key upk^*, a new message m^* and creates a corresponding authentication token π^*, and outputs upk^*, m^*, π^*.

$\mathcal{A}$ wins if:

(1) $\mathsf{Verify}(m^*, \pi^*, \mathsf{MPK}) = 1$;
(2) (upk^*, m^*) is not among the pairs (upk_j, m_i) generated during the query phase.

We denote by $\mathbf{Adv}_{\mathcal{A}}^{Unf}(\lambda) = \Pr[\mathcal{A}\ wins\ the\ game]$ the success probability of adversary $\mathcal{A}$ in winning the unforgeability game.

Definition 3 (*Unforgeability*)**.** An EOLTAA scheme is unforgeable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Unf}(\lambda)$ is negligible.

Anonymity. Anonymity for EOLTAA schemes is defined as the following game between the Challenger C and the Adversary $\mathcal{A}$.

1. $\mathcal{A}$ runs $\mathsf{CSetup}(1^\lambda)$ to create a master key pair (MPK, MSK), and gives MPK to C.
2. C runs $\mathsf{UKeyGen}(1^\lambda)$ to generate two public-secret key pairs (upk_0, usk_0) and (upk_1, usk_1), and gives two public keys upk_0, upk_1 to $\mathcal{A}$. Then, $\mathcal{A}$ returns two corresponding certificates σ_0, σ_1 to C.
3. $\mathcal{A}$ selects a public key $upk_j \in \{upk_0, upk_1\}$ and a message m_i, and asks C to authenticate m_i, but all the queried messages cannot have a common event. C runs $\mathsf{Auth}(m_i, upk_j, usk_j, \sigma_j, \mathsf{MPK})$ and outputs a corresponding authentication token π_i.
4. $\mathcal{A}$ chooses and sends C a new message m^* which is not in the set $\{m_i\}$ and cannot have a common event with queried messages. C randomly picks $b \in \{0, 1\}$, uses (upk_b, usk_b, σ_b) to authenticate m^*, and returns the newly generated authentication token π^* to $\mathcal{A}$. After receiving π^*, $\mathcal{A}$ outputs the guess b'.

$\mathcal{A}$ wins if $b' = b$.

We denote by $\mathbf{Adv}_{\mathcal{A}}^{Anon}(\lambda) = \left|\Pr[\mathcal{A}\ wins\ the\ game] - \frac{1}{2}\right|$ the success probability of adversary $\mathcal{A}$ in winning the anonymity game.

Definition 4 (*Anonymity*)**.** An EOLTAA scheme is anonymous if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Anon}(\lambda)$ is negligible.

Linkability. Linkability for EOLTAA schemes is defined as the following game between the Challenger C and the Adversary $\mathcal{A}$.

1. C first runs $\mathsf{CSetup}(1^\lambda)$ to create a master key pair (MPK, MSK), calls $\mathsf{UKeyGen}(1^\lambda)$ to generate n public-secret key pairs (upk_1, usk_1), $\dots$, (upk_n, usk_n), and gives MPK, $upk_1, \dots, upk_n$ to $\mathcal{A}$.
2. $\mathcal{A}$ runs $\mathsf{UKeyGen}(1^\lambda)$ to create a public-secret key pair (upk, usk), and sends upk to C. C runs $\mathsf{CertGen}(upk, \mathsf{MSK})$ and returns a certificate σ.
3. $\mathcal{A}$ chooses a public key upk_j, a message m_i, and asks C to authenticate m_i. C runs $\mathsf{Auth}(m_i, upk_j, usk_j, \sigma_j, \mathsf{MPK})$ and outputs a corresponding authentication token π_i.
4. $\mathcal{A}$ selects two message m_1^*, m_2^* sharing a common event and creates two corresponding authentication tokens π_1^*, π_2^*, and outputs (m_1^*, π_1^*) and (m_2^*, π_2^*).

$\mathcal{A}$ wins if:

(1) $\mathsf{Verify}(m_i^*, \pi_i^*, \mathsf{MPK}) = 1$ for $i = 1, 2$;
(2) $\mathsf{Link}(m_1^*, m_2^*, \pi_1^*, \pi_2^*) = 0$.

We denote by $\mathbf{Adv}_{\mathcal{A}}^{Link}(\lambda) = \Pr[\mathcal{A}\ wins\ the\ game]$ the success probability of adversary $\mathcal{A}$ in winning the linkability game.

Definition 5 (*Linkability*)**.** An EOLTAA scheme is linkable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Link}(\lambda)$ is negligible.

Traceability. Traceability for EOLTAA schemes is defined as the following game between the Challenger $\mathcal{C}$ and the Adversary $\mathcal{A}$.

1. $\mathcal{C}$ runs $\mathsf{CSetup}(1^\lambda)$ to create a master key pair $(\mathsf{MPK}, \mathsf{MSK})$, and runs $\mathsf{UKeyGen}(1^\lambda)$ to generate n public-secret key pairs (upk_1, usk_1), $\dots, (upk_n, usk_n)$, and gives MPK, $upk_1, \dots, upk_n$ to $\mathcal{A}$.
2. $\mathcal{A}$ first runs $\mathsf{UKeyGen}(1^\lambda)$ to create a public-secret key pair (upk, usk), and sends upk to $\mathcal{C}$. $\mathcal{C}$ will return a certificate σ by running $\mathsf{CertGen}(upk, \mathsf{MSK})$. Then, $\mathcal{A}$ gives $\mathcal{C}$ a public key $upk_i \in \{upk_1, \dots, upk_n\}$. $\mathcal{C}$ runs $\mathsf{CertGen}(upk_i, \mathsf{MSK})$ and returns a certificate σ_i.
3. $\mathcal{A}$ chooses a public key upk_j, a message m_i, and asks $\mathcal{C}$ to authenticate m_i. $\mathcal{C}$ runs $\mathsf{Auth}(m_i, upk_j, usk_j, \sigma_j, \mathsf{MPK})$ and outputs a corresponding authentication token π_i.
4. $\mathcal{A}$ selects two message m_1^*, m_2^* sharing a common event and creates two corresponding authentication tokens π_1^*, π_2^*, and outputs (m_1^*, π_1^*) and (m_2^*, π_2^*).

$\mathcal{A}$ wins if:

(1) $\mathsf{Verify}(m_i^*, \pi_i^*, \mathsf{MPK}) = 1$ for $i = 1, 2$;
(2) $\mathsf{Link}(m_1^*, m_2^*, \pi_1^*, \pi_2^*) = 1$;
(3) $\mathsf{Trace}(m_1^*, m_2^*, \pi_1^*, \pi_2^*) = \perp / upk'$, but $upk' \neq upk$.

We denote by $\mathbf{Adv}_{\mathcal{A}}^{Trace}(\lambda) = \Pr[\mathcal{A}\ wins\ the\ game]$ the success probability of adversary $\mathcal{A}$ in winning the traceability game.

Definition 6 (*Traceability*)**.** An EOLTAA scheme is traceable if for all PPT adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{Trace}(\lambda)$ is negligible.

5. Our Proposed EOLTAA

5.1. Generic construction

Similar to other anonymous authentication schemes, we also utilize the zero-knowledge proof technique to achieve anonymity. Specifically, we apply zk-SNARK to obtain an efficient construction. Since the assurance contributing to linkability-and-traceability is the event, consequently, we will exploit it to support an anonymous-and-accountable requirement. In a nutshell, the authentication process creates a linking tag committed to the event and the user's secret key. To satisfy the requirement of traceability, we will also provide a tracing tag which will be used for tracking.

Let $\mathsf{S} = (\mathsf{S.Setup}, \mathsf{S.Sign}, \mathsf{S.Verify})$ and $\mathsf{U} = (\mathsf{U.Setup}, \mathsf{U.Sign}, \mathsf{U.Verify})$ be two signature schemes, $\mathcal{UPK}, \mathcal{USK}$ be the public key space and secret key space of the signature scheme U. Without loss of generality, the secret key space $\mathcal{USK}$ is defined in a finite field. We assume the public function $\mathcal{F} : \mathcal{USK} \longrightarrow \mathcal{UPK}$ denotes a map between the secret key space and public key space, and satisfies the homomorphic property.[1] Let $\mathsf{Z} = (\mathsf{Z.Setup}, \mathsf{Z.Prover}, \mathsf{Z.Verifier})$ be the zk-SNARK protocol. The EOLTAA scheme is constructed as follows.

- $\mathsf{CSetup}(1^\lambda, \mathcal{L})$. The master key generation algorithm first invokes $\mathsf{S.Setup}(1^\lambda)$ to create a signing key msk and a verification key mpk, and calls $\mathsf{Z.Setup}(1^\lambda, \mathcal{L})$ to obtain the public parameters pp for zk-SNARK, where the language $\mathcal{L}$ is depicted in the following authentication algorithm. It also chooses two hash functions $\mathcal{H} : \{0,1\}^* \times \{0,1\}^* \longrightarrow \mathcal{USK}$ and $\mathcal{H}' : \{0,1\}^* \times \{0,1\}^* \longrightarrow \mathcal{USK}$. The master public key is $\mathsf{MPK} = (mpk, pp, \mathcal{H}, \mathcal{H}')$, the master secret key is $\mathsf{MSK} = msk$.
- $\mathsf{UKeyGen}(1^\lambda)$. The user key generation algorithm calls $\mathsf{U.Setup}(1^\lambda)$ to obtain a secret key $usk_i \in \mathcal{USK}$ and a public key $upk_i \in \mathcal{UPK}$.
- $\mathsf{CertGen}(upk_i, \mathsf{MSK})$. The certificate generation algorithm calls $\mathsf{S.Sign}(upk_i, msk)$ to compute a signature σ_i on a public key upk_i, and outputs σ_i.
- $\mathsf{Auth}(m = e \parallel p, upk_i, usk_i, \sigma_i, \mathsf{MPK})$. On input a message m including an event identifier e and a payload p, the authentication algorithm does the following:
  1. Compute a linking tag $t_1 = \mathcal{H}(e, usk_i)$ and a tracing tag $t_2 = \mathcal{H}'(e, usk_i \parallel upk_i) + p \cdot usk_i$.
  2. Let $x = (m, t_1, t_2, \mathsf{MPK})$ be the common knowledge, $w = (upk_i, usk_i, \sigma_i)$ be the private witness. Call proving algorithm $\mathsf{Z.Prover}(x, w, pp)$ of zk-SNARK for the language $\mathcal{L} = \{x = (m, t_1, t_2, \mathsf{MPK}) \mid \exists w = (upk_i, usk_i, \sigma_i),\ s.t., upk_i = \mathcal{F}(usk_i) \wedge \mathsf{S.Verify}(upk_i, \sigma_i, mpk) = 1 \wedge t_1 = \mathcal{H}(e, usk_i) \wedge t_2 = \mathcal{H}'(e, usk_i \parallel upk_i) + p \cdot usk_i\}$, where $\mathcal{F}$ is to confirm if the two keys correspond to a public-secret key pair, and the S.Verify algorithm is used for checking whether σ_i is a valid certificate. Then Z.Prover algorithm will produce a proof η related to the statement $x \in \mathcal{L}$.
  3. Output an authentication token $\pi = (t_1, t_2, \eta)$.
- $\mathsf{Verify}(m, \pi, \mathsf{MPK})$. The verification algorithm invokes $\mathsf{Z.Verifier}(x, \pi, pp)$, and outputs 0 or 1 for invalid or valid, respectively.
- $\mathsf{Link}(m_1, m_2, \pi_1, \pi_2)$. The link algorithm outputs 1 if two linking tags in π_1, π_2 are the same. Otherwise, it outputs 0.
- $\mathsf{Trace}(m_1, m_2, \pi_1, \pi_2)$. If two different messages m_1, m_2 bind a common event e, the trace algorithm computes a derived secret key usk_i and calls the function $\mathcal{F}$ to calculate the corresponding public key $upk_i = \mathsf{F}(usk_i)$ pointing to the user i.

5.2. Security theorems

We state the following theorems on the security of our construction.

Theorem 1 (*Unforgeability*)**.** *Assume the signature scheme* S *is unforgeable, the function* $\mathcal{F}$ *is one-way and the zk-SNARK scheme* Z *is proof-of-knowledge, then an EOLTAA scheme is unforgeable in the random oracle model.*

Proof. To prove the unforgeability of our EOLTAA scheme, we distinguish the following two types of forgers:

Type-1 forger: In challenge phase, the adversary outputs a three-tuple (upk^*, m^*, π^*) where the upk^* has never been queried by a forger in certificate query phase.

Type-2 forger: In challenge phase, the adversary outputs a three-tuple (upk^*, m^*, π^*) where the upk^* has been queried by a forger in certificate query phase.

We prove this theorem by the following two lemmas.

Lemma 1. *Suppose the signature scheme* S *is unforgeable and the zk-SNARK scheme* Z *is proof-of-knowledge, then the EOLTAA scheme is unforgeable against the Type-1 forger in the random oracle model.*

Proof. Suppose there exists an adversary $\mathcal{A}_E$ that can break the unforgeability of our EOLTAA scheme with non-negligible probability. We build an algorithm $\mathcal{B}$ against the unforgeability of the signature scheme S.

Let $\mathcal{C}_S$ be the challenger corresponding to $\mathcal{B}$ in the game with respect to the signature scheme S. $\mathcal{B}$ runs $\mathcal{A}_E$ to execute the following steps.

The challenger $\mathcal{C}_S$ invokes S.Setup algorithm to create a key pair (mpk, msk), and sends mpk to $\mathcal{B}$. $\mathcal{B}$ calls Z.Setup algorithm to generate parameters pp for zk-SNARK, and selects two hash functions $\mathcal{H}, \mathcal{H}'$. Let $\mathsf{MPK} = (mpk, pp, \mathcal{H}, \mathcal{H}')$. $\mathcal{B}$ runs U.Setup algorithm to create n public and secret key pairs $(upk_1, usk_1), \dots, (upk_n, usk_n)$. Then, $\mathcal{B}$ sends MPK, $upk_1, \dots, upk_n$ to $\mathcal{A}_E$. When $\mathcal{A}_E$ sends a public key $upk_i \in$

[1] For example, the public key is $y = g^x$ (x is the secret key).

$\{upk_1, \ldots, upk_n\}$ to $\mathcal{B}$, $\mathcal{B}$ will send it to C_S. Then C_S runs S.Sign algorithm to generate a corresponding signature σ_i and sends it to $\mathcal{B}$, then $\mathcal{B}$ returns this signature to $\mathcal{A}_E$. And when $\mathcal{A}_E$ sends a public key $upk_j \in \{upk_1, \ldots, upk_n\}$ and a message m_i to $\mathcal{B}$, $\mathcal{B}$ will invoke the simulator algorithm of zk-SNARK to generate a corresponding authentication token π_i, and sends it to $\mathcal{A}_E$. Finally, $\mathcal{A}_E$ outputs upk^*, m^*, π^* where upk^* has never been queried before. $\mathcal{B}$ invokes the extractor algorithm of zk-SNARK to extract the witness (upk^*, usk^*, σ^*) from π^*, and outputs (upk^*, σ^*) as his forgery.

Assume $\mathcal{A}_E$ successfully forges a valid π^*, which will pass verification by Z.Verify, then (upk^*, σ^*) can also pass verification by S.Verify. Since upk^* has never been queried in the certification query phase, then (upk^*, σ^*) is a valid forgery against signature scheme S. Therefore, in this case we would have a successful forger $\mathcal{B}$ against S, contradicting the hypothesis in the statement of the theorem which claims that S is unforgeable. □

Lemma 2. *Suppose the function $\mathcal{F}$ is one-way and the zk-SNARK scheme $\mathcal{Z}$ is proof-of-knowledge, then the EOLTAA scheme is unforgeable against the Type-2 forger in the random oracle model.*

Proof. Suppose there exists an adversary $\mathcal{A}_E$ that can break the unforgeability of our EOLTAA scheme with non-negligible probability. We build an algorithm $\mathcal{B}$ against the one-wayness of function $\mathcal{F}$ with non-negligible probability.

Let C_F be the challenger corresponding to $\mathcal{B}$ in the game with respect to the one-way function $\mathcal{F}$. $\mathcal{B}$ runs $\mathcal{A}_E$ to execute the following steps.

The challenger C_F selects a secret key $x_1 \in \mathcal{USK}$ and computes a corresponding public key y such that the public key has the homomorphic property, and sends y to $\mathcal{B}$. $\mathcal{B}$ invokes S.Setup algorithm to create a key pair (mpk, msk) and calls Z.Setup algorithm to generate parameters pp for zk-SNARK, and selects two hash functions $\mathcal{H}, \mathcal{H}'$. Let $\mathsf{MPK} = (mpk, pp, \mathcal{H}, \mathcal{H}')$. $\mathcal{B}$ also randomly chooses $x_2, \ldots, x_n$ from $\mathcal{USK}$, and sets $upk_1 = y, upk_2 = y \cdot \mathcal{F}(x_2), \ldots, upk_n = y \cdot \mathcal{F}(x_n)$. Then, $\mathcal{B}$ sends $\mathsf{MPK}, upk_1, \ldots, upk_n$ to $\mathcal{A}_E$. When $\mathcal{A}_E$ sends a public key $upk_i \in \{upk_1, \ldots, upk_n\}$ to $\mathcal{B}$, $\mathcal{B}$ will return a corresponding σ_i by running S.Sign algorithm. And when $\mathcal{A}_E$ sends a public key $upk_j \in \{upk_1, \ldots, upk_n\}$ and a message m_i to $\mathcal{B}$, $\mathcal{B}$ will return a corresponding authentication token π_i by calling the simulator algorithm of zk-SNARK. Finally, $\mathcal{A}_E$ outputs upk^*, m^*, π^* where upk^* has been queried before. $\mathcal{B}$ invokes the extractor algorithm of zk-SNARK to recover the witness (upk^*, usk^*, σ^*).

Given upk^* and usk^*, $\mathcal{B}$ can easily obtain the secret x_1 such that $y = \mathcal{F}(x_1)$. This means that $\mathcal{B}$ breaks the one-wayness of $\mathcal{F}$. $\mathcal{B}$ never fails when $\mathcal{A}_E$ succeeds, which concludes the proof. □ □

Theorem 2 (*Anonymity*)**.** *Assume the zk-SNARK scheme $\mathcal{Z}$ is zero-knowledge, then an EOLTAA scheme is anonymous in the random oracle model.*

Proof. Suppose there exists an adversary $\mathcal{A}_E$ that can break the anonymity of our EOLTAA scheme with non-negligible probability. Let C be the challenger corresponding to $\mathcal{A}_E$ in this game.

The adversary $\mathcal{A}_E$ invokes S.Setup algorithm to create a key pair (mpk, msk) and calls Z.Setup algorithm to generate parameters pp for zk-SNARK, and selects two hash functions $\mathcal{H}, \mathcal{H}'$. Let $\mathsf{MPK} = (mpk, pp, \mathcal{H}, \mathcal{H}')$. $\mathcal{A}_E$ sends MPK to C. C selects two secret keys usk_0, usk_1 and calls $\mathcal{F}$ to obtain the corresponding public keys upk_0, upk_1. When C sends upk_0, upk_1 to $\mathcal{A}_E$, $\mathcal{A}_E$ will return σ_0, σ_1 by running S.Sign algorithm. And when $\mathcal{A}_E$ sends a public key $upk_j \in \{upk_0, upk_1\}$ and a message m_i to C, C will return a corresponding authentication token π_i by calling the simulator algorithm of zk-SNARK. Next, $\mathcal{A}_E$ sends a message m^* to C, C will randomly pick $b \in \{0, 1\}$ and use (upk_b, usk_b, σ_b) to create a corresponding authentication token π^*, and sends π^* to $\mathcal{A}_E$. Finally, $\mathcal{A}_E$ outputs the guess b'.

The information of b only exists in the witness (upk_b, usk_b, σ_b) which is used to generate the authentication token π^*. Since π^* is a zero-knowledge proof that will not leak any information about the witness. Thus, the advantage that $\mathcal{A}_E$ can guess b correctly is always $\frac{1}{2}$. □

Theorem 3 (*Linkability*)**.** *Assume the zk-SNARK scheme $\mathcal{Z}$ is zero-knowledge, then an EOLTAA scheme is linkable in the random oracle model.*

Proof. Suppose there exists an adversary $\mathcal{A}_E$ that can break the linkability of our EOLTAA scheme with non-negligible probability. Let C be the challenger corresponding to $\mathcal{A}_E$ in this game.

The challenger C runs S.Setup algorithm to create a key pair (mpk, msk) and invokes Z.Setup algorithm to generate parameters pp for zk-SNARK, and selects two hash functions $\mathcal{H}, \mathcal{H}'$. Let $\mathsf{MPK} = (mpk, pp, \mathcal{H}, \mathcal{H}')$. C runs U.Setup to create n key pairs $(upk_1, usk_1), \ldots, (upk_n, usk_n)$, and sends $\mathsf{MPK}, upk_1, \ldots, upk_n$ to $\mathcal{A}_E$. $\mathcal{A}_E$ first calls U.Setup to generate a key pair (upk_A, usk_A), and sends upk_A to C. C will return a corresponding σ_A by running S.Sign algorithm. When $\mathcal{A}_E$ sends a public key $upk_j \in \{upk_1, \ldots, upk_n\}$ and a message m_i to C, C will return an authentication token π_i by calling the simulator algorithm of zk-SNARK. Finally, $\mathcal{A}_E$ selects two messages m_1^*, m_2^* sharing a common event, generates the corresponding π_1^*, π_2^*, and outputs $(m_1^*, \pi_1^*), (m_2^*, \pi_2^*)$.

Since the zk-SNARK is zero-knowledge, $\mathcal{A}_E$ cannot get the information about $usk_1, \ldots, usk_n$ and the corresponding $\sigma_1, \ldots, \sigma_n$ in the authentication query phase. If π_1^*, π_2^* can pass verification by Z.Verify, then $\mathcal{A}_E$ must use usk_A to generate the tag t_1. Since t_1 is calculated by $\mathcal{H}(e, usk_i)$ and usk_A is the same one, thus the two linking tags in π_1^*, π_2^* are equal, and the π_1^*, π_2^* are linked. This contradicts with our assumption that $\mathcal{A}_E$ can break the linkability. □

Theorem 4 (*Traceability*)**.** *Assume the zk-SNARK scheme $\mathcal{Z}$ is proof-of-knowledge, then an EOLTAA scheme is traceable in the random oracle model.*

Proof. Suppose there exists an adversary $\mathcal{A}_E$ that can break the traceability of our EOLTAA scheme with non-negligible probability. Let C be the challenger corresponding to $\mathcal{A}_E$ in this game.

The challenger C invokes S.Setup algorithm to create a key pair (mpk, msk) and calls Z.Setup algorithm to generate parameters pp for zk-SNARK, and selects two hash functions $\mathcal{H}, \mathcal{H}'$. Let $\mathsf{MPK} = (mpk, pp, \mathcal{H}, \mathcal{H}')$. C also generates n key pairs $(upk_1, usk_1), \ldots, (upk_n, usk_n)$ by calling U.Setup algorithm. Then, C sends $\mathsf{MPK}, upk_1, \ldots, upk_n$ to $\mathcal{A}_E$. Next, $\mathcal{A}_E$ runs U.Setup to create a key pair (upk_A, usk_A), and sends upk_A to C. C will return a corresponding σ_A by running S.Sign algorithm. When $\mathcal{A}_E$ sends a public key $upk_i \in \{upk_1, \ldots, upk_n\}$ to C, C will call S.Sign algorithm to return a corresponding σ_i. And when $\mathcal{A}_E$ sends a public key $upk_j \in \{upk_1, \ldots, upk_n\}$ and a message m_i to C, C will return an authentication token π_i by calling the simulator algorithm of zk-SNARK. Finally, $\mathcal{A}_E$ selects two messages m_1^*, m_2^* sharing a common event, and generates the corresponding π_1^*, π_2^*, and outputs $(m_1^*, \pi_1^*), (m_2^*, \pi_2^*)$.

If π_1^*, π_2^* can pass verification by Z.Verify, then we can call the extractor algorithm of zk-SNARK to extract the corresponding witness (upk_A, usk_A, σ_A), and upk_A can also be easily obtained by invoking the function $\mathcal{F}$ according to usk_A. Since we have proved that our EOLTAA scheme is unforgeable, thus $\mathcal{A}_E$ cannot forge a valid certificate. However, $\mathcal{A}_E$ only has (upk_A, usk_A, σ_A), so π_1^*, π_2^* must be generated by using (upk_A, usk_A, σ_A). Therefore, the advantage that $\mathcal{A}_E$ can break the traceability is negligible. □

5.3. Practical instantiation

We instantiate S and U using the Schnorr signature [59] as the signature scheme, which is used to create a user's certificate and key pair, and instantiate Z using the implementation proposed by Ben-Sasson et al. [60], respectively.

Table 2
Functionality comparison of different schemes.

Scheme	[14]	[6]	[20]	[21]	Ours
Anonymity	✓	✓	✓	✓	✓
Linkability	✓	✓	✓	✓	✓
Traceability	×	✓	✓	✓	✓

Let G be a group of prime order q and g be a generator. Let $\mathcal{H}_1 : \{0,1\}^* \longrightarrow \mathbb{Z}_q$, $\mathcal{H}_2 : \{0,1\}^* \times \{0,1\}^* \longrightarrow \mathbb{Z}_q$ be two hash functions. Pick a secret key $x \in \mathbb{Z}_q$ as the signing key and compute $y = g^x$, and set $(G, q, g, y, \mathcal{H}_1)$ as the verification key of the signature algorithm. For a user, he first picks a secret key $x_i \in \mathbb{Z}_q$ and generates a public key $y_i = g^{x_i}$. To issue a certificate, randomly select $r \in \mathbb{Z}_q$ and calculate $h = g^r$, and compute $s = r + cx$ where $c = \mathcal{H}_1(y_i \parallel h)$, and set $\sigma_i = (c, s)$ as a certificate for the user with a public key y_i.

To create an authentication token, a user chooses $e \in \mathbb{Z}_q, p \in \mathbb{Z}_q$ to form a message $e \parallel p$, and generates two tags $t_1 = \mathcal{H}_2(e, x_i)$, $t_2 = \mathcal{H}_2(e, x_i \parallel y_i) + p \cdot x_i$. The user with the witness (x_i, y_i, c, s) only needs to prove that the relation $y_i = g^{x_i} \wedge g^s = h \cdot y^c \wedge t_1 = \mathcal{H}_2(e, x_i) \wedge t_2 = \mathcal{H}_2(e, x_i \parallel y_i) + p \cdot x_i$ holds. Specifically, to prove $upk_i = \mathcal{F}(usk_i)$, the user only needs to certify that $y_i = g^{x_i}$; to prove $\mathsf{S.Verify}(upk_i, \sigma_i, mpk) = 1$, the user shows that $g^s = h \cdot y^c$.

We show that the correctness of our instantiation. First, it is obvious that the verification algorithm is correct. We have a key pair (x_i, y_i) such that $y_i = g^{x_i}$ and possess a certificate (c, s), where $c = \mathcal{H}_1(y_i \parallel h)$ and $s = r + cx$, and only to compute $h = g^s y^{-c}$ and verify if c is equal to $\mathcal{H}_1(y_i \parallel h)$. The link algorithm is correct as follows. Given two valid authentication tokens $\pi_1 = (t_1^1, t_2^1, \eta^1)$ and $\pi_2 = (t_1^2, t_2^2, \eta^2)$ and the same event identifier e, we have the following relations: $t_1^1 = \mathcal{H}_2(e, x_i) = t_1^2$. Finally, we show that the trace algorithm correctly works. Given two linked authentication tokens $\pi_1 = (t_1^1, t_2^1, \eta^1)$ and $\pi_2 = (t_1^2, t_2^2, \eta^2)$, where $t_1^1 = \mathcal{H}_2(e, x_i)$, $t_2^1 = \mathcal{H}_2(e, x_i \parallel y_i) + p_1 \cdot x_i$, $t_1^2 = \mathcal{H}_2(e, x_i)$, and $t_2^2 = \mathcal{H}_2(e, x_i \parallel y_i) + p_2 \cdot x_i$, we have $x_i = \frac{t_2^2 - t_2^1}{p_2 - p_1}$.

5.4. Implementation

We implement our EOLTAA scheme based on libsnark [61], which is a library that implements zk-SNARKs in C++. We analyze the performance of our scheme as follows.

Firstly, we will discuss the functionalities of our EOLTAA scheme and make a comparison with some recent anonymous authentication schemes achieving linking and/or tracing, such as linkable ring signatures [14] and linkable group signatures [20,21] and linkable anonymous credentials [6], as shown in Table 2. As all of these schemes hold the basic anonymity, we mainly pay attention to linkability and traceability. [14] ensures the linkability without the traceability, while [6], [20] and [21] achieve the linkability and traceability simultaneously. As shown in Section 5, we have proved that our EOLTAA scheme has satisfied the security requirements, i.e. our scheme also supports the basic linkability and traceability.

Secondly, we will analyze the efficiency of our EOLTAA scheme. We evaluate the performance of EOLTAA on a computer with AMD Ryzen 5 3600 6-core processor running Ubuntu 20.04. We set the security parameter $\lambda = 160$, and use the SHA-256 as the secure hash function. Next, we select the recent anonymous authentication schemes with constant size, and demonstrate the performance in terms of running time, size, storage and communication cost in Tables 3 and 4. Note that, these above group signature and ring signature schemes can also be reviewed as authentication schemes in some degree, thus we regard a signature as an authentication on a message below.

(1) Running Time: Now, we first analyze the running time of the authentication process. To authenticate a message, we should use the libsnark library to generate a non-interactive zero-knowledge proof, thus we list the running time of the Z.Prover algorithm and Z.Verifier algorithm of the zk-SNARK. We use the Z.Prover algorithm to produce a non-interactive proof, and the proof generation time is 970.23 ms. Then, we verify the proof by calling the Z.Verifier algorithm, and the verification time is 5.34 ms. As shown in Table 3, we also list the running time of authentication generation and verification of other related schemes. It can be learned from the table that our EOLTAA scheme has a slightly longer authentication time, but with a faster verification time than others.

Table 3
Running time comparison of different schemes.

Scheme	[6]	[20]	[21]	Ours
Authentication (ms)	93.64	21.38	68.22	970.23
Verification (ms)	89.17	42.92	12.29	5.34

Table 4
Size, storage and communication cost comparison of different schemes.

Scheme	[6]	[20]	[21]	Ours
Authentication size (kB)	0.80	0.17	1.30	0.12
Storage cost (kB)	0.81	0.18	1.32	0.19
Communication cost (kB)	0.81	0.18	1.32	0.19

(2) Size, Storage and Communication Cost: Now, we will analyze the size, storage and communication cost of the authentication process. We make use of the zk-SNARK to create a non-interactive proof for authentication. Due to the succinctness of zk-SNARK, the size of authentication on a message is 0.12 kB, which is the minimum length in these schemes in Table 4. Then, a user creates a token, i.e. $\pi = (t_1, t_2, \eta)$, and submits the message and authentication token pair (m, π), thus the communication cost is 0.19 kB. Since an anonymous authentication on a message should be saved as (m, π) for linking and tracing, the storage cost is also 0.19 kB. It can be learned from the table that our EOLTAA scheme has a minimum length of authentication, and with a slightly smaller storage and communication cost.

From the above comparisons and analyses, we can find that our EOLTAA scheme satisfies all the security requirements. The simulation results show that our EOLTAA scheme is feasible and efficiently calculated.

6. An anonymous and traceable voting protocol based on blockchain

Voting plays an important part in modern democracies, and a voting activity is the most straightforward fashion to execute the populace's right. In fact, a ballot represent a preference for the candidate that the voter supported. Obviously, the integrity of voting process is of vital importance to the integrity of the democracy itself [62]. Even through various cryptographic techniques have been exploited to realize anonymous voting, most traditional e-voting schemes cannot guarantee the transparency and the security against tampering. Fortunately, many solutions based on blockchain have been proposed to efficiently eliminate this issue. In this section, we present an anonymous and traceable e-voting protocol based on blockchain using our proposed EOLTAA scheme.

Traditional e-voting systems mainly need a public bulletin board to realize a communication channel and store information as a database [63] which may suffer from data loss and modification. Centralized voting platforms, such as [64] and [65], always suffer from single point of failure, and are easily attacked from adversaries such that privacy leaks always happen. Some self-tallying voting protocols, like [66] and [67], can realize decentralized property, but they are unsuitable for large scale voting, because even single one that does not join in the protocol will lead to the failure of tallying.

Table 5
Comparison among the existing blockchain-based voting schemes.

Scheme	Decentralized	Anonymous	Double-voting avoided	Linkable	Traceable
[58,69,70]	✓	✓	×	×	×
[74,75]	✓	×	✓	✓	×
[73,76]	✓	✓	✓	✓	×
Ours	✓	✓	✓	✓	✓

Fortunately, blockchain technology is used in voting to reduce the dependence on centralized system and avoid single point of failure, data modification and loss. A comparison among the existing blockchain-based voting schemes is showed in Table 5. In [58] and [68–71], all the ballots are submitted and recorded on blockchain, which just serves as a public and immutable database. As a matter of fact, double voting is always a challenge for most e-voting schemes. In order to avoid double voting, solutions in [72–76] are designed to eliminate this risk. [73] and [76] use linkable ring signature to achieve the linkability of two ballots that correspond to the same voter, but they lack of the traceability.

All the aforementioned schemes are not applicable for wide adoption because they cannot efficiently solve the double voting. And no novel manners are proposed to simultaneously detect double voting and locate the double voter. Moreover, a common disadvantage of these schemes is that they cannot provide decentralization, double-voting resistance, linkability and traceablity at the same time. However, using EOLTAA, we can construct the first decentralized, anonymous, linkable and publicly traceable e-voting scheme based on blockchain to meet this need.

6.1. Participants

There are four entities that participate in the voting as shown in Fig. 1, which consists of election committee (EC), certificate authority (CA), blockchain voting platform and voters.

Election Committee: An election committee is a voting organizer who acts as vote initiator and counter and is responsible for publishing voting activity and computing the final election result.

Certificate Authority: A certificate authority is a trusted authority whose duty is to issue certificates to participants who want to obtain a certificate.

Blockchain Voting Platform: A blockchain voting platform acts as a public bulletin board, which stores some public announcements and all voters' ballots.

Voters: A set of voters should register a certificate and receive a voting announcement, and cast their ballots to the blockchain platform.

6.2. Security requirements

Informally, we wish an e-voting scheme should satisfy a number of security requirements.

Privacy: The content of each ballot should be protected such that anyone cannot obtain the real information included in the ballot.

Anonymity: All valid voters who follow the voting protocol are kept secret.

Eligibility: Only authorized voters are allowed to vote.

Verifiability: Any party, including a bystander, can be convinced that all valid ballots are included in the final result when tallying.

Unforgeability: An attacker cannot forge a voter's identity to cast a ballot in the name of the voter.

Traceability: Double voting can be detected, and it is possible to further infer the double voter's identity without the help of other parties.

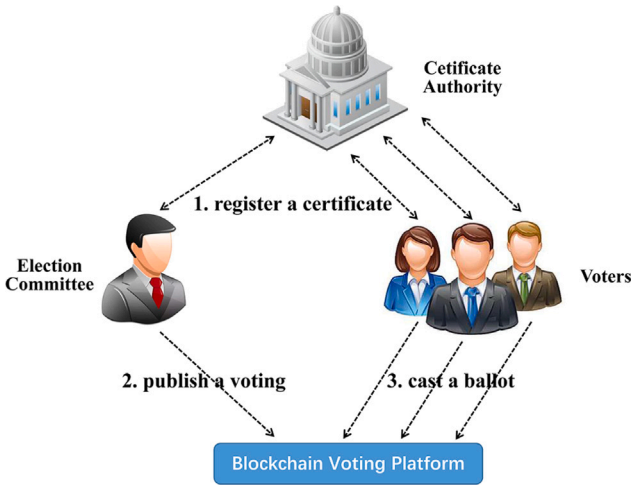

Fig. 1. The system model of e-voting.

6.3. The voting protocol

Let $\Phi = (\Phi.\mathsf{CSetup}, \Phi.\mathsf{UKeyGen}, \Phi.\mathsf{CertGen}, \Phi.\mathsf{Auth}, \Phi.\mathsf{Verify}, \Phi.\mathsf{Link}, \Phi.\mathsf{Trace})$ be an event-oriented linkable and traceable anonymous authentication scheme, and let $\mathsf{E} = (\mathsf{E.KeyGen}, \mathsf{E.Encrypt}, \mathsf{E.Decrypt})$ be a public key encryption scheme that is semantically secure. We now construct an anonymous and traceable e-voting protocol, which consists of the following four phases: setup phase, registration phase, voting phase and counting phase.

Setup: *Before starting registration, setup phase is required to initiate a voting.*

CA first prepares the master public key MPK by running $\Phi.\mathsf{CSetup}$ algorithm, and sends a transaction including the public parameters to the blockchain network.

Registration: *Voters and the election committee should generate their key pairs and register a certificate with the certificate authority.*

EC runs $\Phi.\mathsf{UKeyGen}$ algorithm to create a public and secret key pair (pk_{EC}, sk_{EC}), and obtains a certificate σ_{EC} from CA by calling $\Phi.\mathsf{CertGen}$ algorithm. Similarly, a voter V_i creates a public and secret key pair (upk_i, usk_i) and obtains a certificate σ_i.

Voting: *The election committee publishes a voting, and each voter anonymously casts a ballot.*

EC picks a random number vid defined as the identifier of a voting activity, calls $\mathsf{E.KeyGen}$ to create a key pair (pk_e, sk_e) used for encrypting the ballot, and sets a deadline t, a candidate list cl and a counting policy Y. Then, EC creates a new blockchain account address $addr_{EC}$, and generates $\pi_{EC} = \Phi.\mathsf{Auth}(vid \parallel addr_{EC}, pk_{EC}, sk_{EC}, \sigma_{EC}, \mathsf{MPK})$ to authenticate $vid \parallel addr_{EC}$. At last, EC codes a smart contract sc that includes all above vid, π_{EC}, pk_e, cl, t, Y, MPK, etc. in this voting, and sends $(vid \parallel addr_{EC}, sc, \pi_{EC})$ to the blockchain using $addr_{EC}$.

When receiving this voting, voter V_i chooses one candidate and runs $\mathsf{E.Encrypt}$ to encrypt it under the public key pk_e to get C_i, and calls $\Phi.\mathsf{Auth}$ algorithm to generate $\pi_i = \Phi.\mathsf{Auth}(vid \parallel C_i, upk_i, usk_i, \sigma_i, \mathsf{MPK})$, and uses his newly created account address $addr_i$ to send (C_i, π_i) to the blockchain network.

Counting: *The election committee opens the encrypted ballots and calculates the final election result.*

Smart contract sc runs $\Phi.\mathsf{Verify}(vid \parallel C_i, \pi_i, \mathsf{MPK})$ to verify each received ballot and authentication pair, and removes invalid ones. Then, it starts to detect double voting by running $\Phi.\mathsf{Link}(C_i, C_*, \pi_i, \pi_*)$ for each valid (C_*, π_*) that has passed verification, and executes $\Phi.\mathsf{Trace}(C_i, C_*, \pi_i, \pi_*)$ to further derive the double voter's identity.

Then, EC collects all valid ballots which exclude these corresponding to double voting and invalid authentications, and runs $\mathsf{E.Decrypt}$

to decrypt ballots by using sk_e and calculates the final election result $result$. Moreover, EC also generates a zero knowledge proof π_{result}, with the secret key sk_e as the witness. In particular, the proof is for the language $\mathcal{L}_{result} = \{Para \mid \exists sk_e \, s.t., pk_e = f(sk_e) \wedge_{i=1}^{n} b_i = \mathsf{E.Decrypt}(sk_e, C_i) \wedge result = Y(b_i; b_1, \ldots, b_n)\}$, where $Para$ denotes the public parameters together with the ciphertexts $C_1, \ldots, C_n$, b_i is the plaintext corresponding to C_i, and f is the function to verify if the two keys correspond to a key pair. At the end, EC sends $(result, \pi_{result})$ to the blockchain network, and anyone can check the final election result by verifying the proof π_{result}.

6.4. Security analysis

We discuss the security our proposed voting protocol satisfied.

Privacy: All the ballots recorded on the blockchain are encrypted by the key pk_e of the public key encryption scheme E. We assume the encryption scheme is semantically secure, thus the encrypted ballots will not leak information to the public. And only the election committee who acts as the initiator and has the secret key sk_e can decrypt them.

Anonymity: A randomly generated blockchain address $addr_i$ eliminates the risk of locating a certain voter through several transactions he has ever involved. Since the Φ scheme is anonymous, η is a zero-knowledge proof that leaks nothing useful information, and t_1, t_2 in the generated authentication token π_i correspond to two random numbers, i.e., the proof and tokens reveal nothing about a certain voter's identity information. Furthermore, any one, including the certificate authority and the election committee, cannot distinguish which ballot corresponds to which voter.

Eligibility: A legitimate voter should register at the certificate authority and obtain a corresponding certificate binding his identity information. Then he can create a valid authentication to attest his validity, and everyone can verify the correctness. However, for an illegitimate user who did not obtain a certificate from the certificate authority, he cannot create a valid authentication without an indispensable certificate.

Verifiability: The whole voting process is showed on the blockchain voting platform, from which anyone is capable of checking all the ballots and the corresponding authentication tokens, and of verifying whether they are valid or not by running Φ.Verify algorithm. And the final election result $result$ can be verified by checking the validity of the proof π_{result}.

Unforgeability: Each ballot–token pair (C_i, π_i) recorded on the blockchain is visible for the public, but both the encrypted ballot and the token are the same as randomly generated numbers. The Φ scheme is unforgeable, so it is impossible to forge a voter's private information, such as secret key and certificate, to create a valid authentication token that can be successfully accepted through Φ.Verify in the name of the voter.

Traceability: In a voting activity with a fixed identifier vid, every voter is allowed to cast one ballot together with a token, which includes a linking tag t_1 and a tracing tag t_2. Φ scheme is linkable and traceable, so if a voter tries to cast twice, the two linking tags will lead to the linkability of the two tokens since the same hash value of $\mathcal{H}(vid, usk_i)$. Therefore, we can accurately locate the origin of double voting. Furthermore, with the help of Φ scheme, two tracing tags in the two tokens are designed to reveal the identity of this double voter. That is to say, Φ scheme provides a novel solution in which double voting can be detected and the anonymity is further removable through deriving the double voter's identity without the help of other parties.

7. Conclusion

In this paper, we propose a new variant of anonymous authentication called event-oriented linkable and traceable anonymous authentication that simultaneously achieves unforgeability, anonymity, linkability and traceability. All these properties of our scheme are proven secure. Then, we give a generic construction to satisfy the security requirements. In particular, we use this variant to construct the first decentralized, anonymous, linkable and traceable e-voting based on blockchain, and analysis the security of our e-voting scheme.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to sincerely thank all the reviewers for your time and expertise on this work. Your insightful comments help us improve this work. This work was supported by the National Natural Science Foundation of China (Grant Nos. U2001205, 61922036, 61932011), and the Guangdong Basic and Applied Basic Research Foundation, China (Grant Nos. 2019B030302008, 2019B1515120010).

References

[1] Tsang PP, Smith SW. PPAA: Peer-to-peer anonymous authentication. In: International Conference on Applied Cryptography and Network Security. Springer; 2008, p. 55–74.
[2] Camenisch J, Drijvers M, Lehmann A. Universally composable direct anonymous attestation. In: Public-Key Cryptography–PKC 2016. Springer; 2016, p. 234–64.
[3] Krenn S, Samelin K, Striecks C. Practical group-signatures with privacy-friendly openings, In: Proceedings of the 14th International Conference on Availability, Reliability and Security, 2019: p. 1–10.
[4] Zheng H, Wu Q, Qin B, Zhong L, He S, Liu J. Linkable group signature for auditing anonymous communication. In: Australasian Conference on Information Security and Privacy. Springer; 2018, p. 304–21.
[5] Boyen X, Haines T. Forward-secure linkable ring signatures. In: Australasian Conference on Information Security and Privacy. Springer; 2018, p. 245–64.
[6] Diao F, Zhang F, Cheng X. A privacy-preserving smart metering scheme using linkable anonymous credential. IEEE Trans Smart Grid 2014;6(1):461–7.
[7] Bootle J, Cerulli A, Chaidos P, Ghadafi E, Groth J, Petit C. Short accountable ring signatures based on DDH. In: European Symposium on Research in Computer Security. Springer; 2015, p. 243–65.
[8] El Kaafarani A, Ghadafi E, Khader D. Decentralized traceable attribute-based signatures. In: Cryptographers' Track at the RSA Conference. Springer; 2014, p. 327–48.
[9] Gu K, Wang K, Yang L. Traceable attribute-based signature. J Inform Secur Appl 2019;49:102400.
[10] Blömer J, Bobolz J, Diemert D, Eidens F. Updatable anonymous credentials and applications to incentive systems, In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019, pp. 1671–1685.
[11] Camenisch J, Lehmann A, Neven G, Rial A. Privacy-preserving auditing for attribute-based credentials. In: European Symposium on Research in Computer Security. Springer; 2014, p. 109–27.
[12] Maji HK, Prabhakaran M, Rosulek M. Attribute-based signatures. In: Cryptographers' Track at the RSA Conference. Springer; 2011, p. 376–92.
[13] Urquidi M, Khader D, Lancrenon J, Chen L. Attribute-based signatures with controllable linkability. In: International Conference on Trusted Systems. Springer; 2015, p. 114–29.
[14] Wang X, Chen Y, Ma X. Adding linkability to ring signatures with one-time signatures. In: International Conference on Information Security. Springer; 2019, p. 445–64.
[15] El Kaafarani A, Chen L, Ghadafi E, Davenport J. Attribute-based signatures with user-controlled linkability. In: International Conference on Cryptology and Network Security. Springer; 2014, p. 256–69.
[16] Kaaniche N, Laurent M. Attribute-based signatures for supporting anonymous certification. In: European Symposium on Research in Computer Security. Springer; 2016, p. 279–300.
[17] Kuchta V, Sahu RA, Saraswat V, Sharma G, Sharma N, Markowitch O. Anonymous yet traceable strong designated verifier signature. In: International Conference on Information Security. Springer; 2018, p. 403–21.
[18] Lueks W, Alpár G, Hoepman J-H, Vullers P. Fast revocation of attribute-based credentials for both users and verifiers. Comput Secur 2017;67:308–23.
[19] Derler D, Slamanig D. Highly-efficient fully-anonymous dynamic group signatures, In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, 2018: p. 551–565.
[20] Hwang JY, Chen L, Cho HS, Nyang D. Short dynamic group signature scheme supporting controllable linkability. IEEE Trans Inf Forensics Secur 2015;10(6):1109–24.

[21] Zheng H, Wu Q, Guan Z, Qin B, He S, Liu J. Achieving liability in anonymous communication: Auditing and tracing. Comput Commun 2019;145:1–13.

[22] Tsang PP, Wei VK, Chan TK, Au MH, Liu JK, Wong DS. Separable linkable threshold ring signatures. In: International Conference on Cryptology in India. Springer; 2004, p. 384–98.

[23] Au MH, Liu JK, Susilo W, Yuen TH. Secure ID-based linkable and revocable-iff-linked ring signature with constant-size construction. Theoret Comput Sci 2013;469:1–14.

[24] Au MH, Susilo W, Yiu S-M. Event-oriented k-times revocable-iff-linked group signatures. In: Australasian Conference on Information Security and Privacy. Springer; 2006, p. 223–34.

[25] Fujisaki E. Sub-linear size traceable ring signatures without random oracles. In: Cryptographers' Track At the RSA Conference. Springer; 2011, p. 393–415.

[26] Wei VK. Tracing-by-linking group signatures. In: International Conference on Information Security. Springer; 2005, p. 149–63.

[27] Brickell E, Camenisch J, Chen L. Direct anonymous attestation, In: Proceedings of the 11th ACM Conference on Computer and Communications Security, 2004: p. 132–145.

[28] Barki A, Desmoulins N, Gharout S, Traoré J. Anonymous attestations made practical, In: Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2017: p. 87–98.

[29] Yang G, Wong DS, Deng X, Wang H. Anonymous signature schemes. In: International Workshop on Public Key Cryptography. Springer; 2006, p. 347–63.

[30] Fischlin M. Anonymous signatures made easy. In: International Workshop on Public Key Cryptography. Springer; 2007, p. 31–42.

[31] Rivest RL, Shamir A, Tauman Y. How to leak a secret. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer; 2001, p. 552–65.

[32] Yang X, Wu W, Liu JK, Chen X. Lightweight anonymous authentication for ad hoc group: A ring signature approach. In: International Conference on Provable Security. Springer; 2015, p. 215–26.

[33] Au MH, Liu JK, Susilo W, Zhou J. Realizing fully secure unrestricted ID-based ring signature in the standard model based on HIBE. IEEE Trans Inform Forensics Secur 2013;8(12):1909–22.

[34] Wang W, Chen S. Attribute-based ring signature scheme with constant-size signature. IET Inf Secur 2010;4(2):104–10.

[35] Deng L, Zeng J. Two new identity-based threshold ring signature schemes. Theoret Comput Sci 2014;535:38–45.

[36] Bellare M, Fuchsbauer G. Policy-based signatures. In: International Workshop on Public Key Cryptography. Springer; 2014, p. 520–37.

[37] Okamoto T, Takashima K. Decentralized attribute-based signatures. In: International Workshop on Public Key Cryptography. Springer; 2013, p. 125–42.

[38] Tan S, Groß T. Monipoly —- an expressive q-SDH-based anonymous attribute-based credential system. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer; 2020, p. 498–526.

[39] Garman C, Green M, Miers I. Decentralized anonymous credentials.. In: NDSS. Citeseer; 2014.

[40] Baldimtsi F, Lysyanskaya A. Anonymous credentials light, In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, 2013: p. 1087–1098.

[41] Crites EC, Lysyanskaya A. Delegatable anonymous credentials from mercurial signatures. In: Cryptographers' Track at the RSA Conference. Springer; 2019, p. 535–55.

[42] Deuber D, Maffei M, Malavolta G, Rabkin M, Schröder D, Simkin M. Functional credentials. Proc Privacy Enhancing Technol 2018;2018(2):64–84.

[43] Yang R, Au MH, Xu Q, Yu Z. Decentralized blacklistable anonymous credentials with reputation. Comput Secur 2019;85:353–71.

[44] Nakanishi T, Fujiwara T, Watanabe H. A linkable group signature and its application to secret voting. Trans Inform Process Soc Japan 1999;40(7):3085–96.

[45] Lu Y, Tang Q, Wang G. Zebralancer: Private and anonymous crowdsourcing system atop open blockchain. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE; 2018, p. 853–65.

[46] Ateniese G, Camenisch J, Joye M, Tsudik G. A practical and provably secure coalition-resistant group signature scheme. In: Annual International Cryptology Conference. Springer; 2000, p. 255–70.

[47] Slamanig D, Spreitzer R, Unterluggauer T. Adding controllable linkability to pairing-based group signatures for free. In: International Conference on Information Security. Springer; 2014, p. 388–400.

[48] Zhang X, Liu JK, Steinfeld R, Kuchta V, Yu J. Revocable and linkable ring signature. In: International Conference on Information Security and Cryptology. Springer; 2019, p. 3–27.

[49] Teranishi I, Furukawa J, Sako K. K-times anonymous authentication. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer; 2004, p. 308–22.

[50] Au MH, Liu JK, Susilo W, Yuen TH. Constant-size ID-based linkable and revocable-iff-linked ring signature. In: International Conference on Cryptology in India. Springer; 2006, p. 364–78.

[51] Fujisaki E, Suzuki K. Traceable ring signature. In: International Workshop on Public Key Cryptography. Springer; 2007, p. 181–200.

[52] Zhou J, Lam KY. Securing digital signatures for non-repudiation. Comput Commun 1999;22(8):710–6.

[53] Boneh D, Shen E, Waters B. Strongly unforgeable signatures based on computational Diffie-Hellman. In: International Workshop on Public Key Cryptography. Springer; 2006, p. 229–40.

[54] Dwork C, Naor M. An efficient existentially unforgeable signature scheme and its applications. J Cryptol 1998;11(3):187–208.

[55] Groth J. Short pairing-based non-interactive zero-knowledge arguments. In: International Conference on the Theory and Application of Cryptology and Information Security. Springer; 2010, p. 321–40.

[56] Groth J, Kohlweiss M, Maller M, Meiklejohn S, Miers I. Updatable and universal common reference strings with applications to zk-SNARKs. In: Annual International Cryptology Conference. Springer; 2018, p. 698–728.

[57] Sasson EB, Chiesa A, Garman C, Green M, Miers I, Tromer E, Virza M. Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. IEEE; 2014, p. 459–74.

[58] Zhao Z, Chan THH. How to vote privately using bitcoin. In: International Conference on Information and Communications Security. Springer; 2015, p. 82–96.

[59] Schnorr C-P. Efficient identification and signatures for smart cards. In: Conference on the Theory and Application of Cryptology. Springer; 1989, p. 239–52.

[60] Ben-Sasson E, Chiesa A, Tromer E, Virza M. Succinct non-interactive zero knowledge for a von Neumann architecture. In: 23rd {USENIX} Security Symposium. 2014, p. 781–96.

[61] Ben-Saason E, Chiesa A, Genkin D, Kfir S, Tromer E, Virza M. Libsnark: a c++ library for zksnark proofs. 2014, https://github.com/scipr-lab/libsnark.

[62] Kohno T, Stubblefield A, Rubin AD, Wallach DS. Analysis of an electronic voting system. In: IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004. IEEE; 2004, p. 27–40.

[63] Sampigethaya K, Poovendran R. A framework and taxonomy for comparison of electronic voting schemes. Comput Ssecur 2006;25(2):137–53.

[64] Adida B. Helios: Web-based open-audit voting. In: USENIX Security Symposium, Vol. 17. 2008, p. 335–48.

[65] Culnane C, Schneider S. A peered bulletin board for robust use in verifiable voting systems. In: 2014 IEEE 27th Computer Security Foundations Symposium. IEEE; 2014, p. 169–83.

[66] Kiayias A, Yung M. Self-tallying elections and perfect ballot secrecy. In: International Workshop on Public Key Cryptography. Springer; 2002, p. 141–58.

[67] Groth J. Efficient maximal privacy in boardroom voting and anonymous broadcast. In: International Conference on Financial Cryptography. Springer; 2004, p. 90–104.

[68] Bellini E, Ceravolo P, Damiani E. Blockchain-based e-vote-as-a-service. In: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD). IEEE; 2019, p. 484–6.

[69] Li Y, Susilo W, Yang G, Yu Y, Liu D, Du X, Guizani M. A blockchain-based self-tallying voting protocol in decentralized IoT. IEEE Trans Dependable Secure Comput 2020.

[70] McCorry P, Shahandashti SF, Hao F. A smart contract for boardroom voting with maximum voter privacy. In: International Conference on Financial Cryptography and Data Security. Springer; 2017, p. 357–75.

[71] Zhang W, Yuan Y, Hu Y, Huang S, Cao S, Chopra A, Huang S. A privacy-preserving voting protocol on blockchain. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). IEEE; 2018, p. 401–8.

[72] Dimitriou T. Efficient, coercion-free and universally verifiable blockchain-based voting. Comput Netw 2020;174:107234.

[73] Lyu J, Jiang ZL, Wang X, Nong Z, Au MH, Fang J. A secure decentralized trustless E-voting system based on smart contract. In: 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE). IEEE; 2019, p. 570–7.

[74] Yang X, Yi X, Nepal S, Han F. Decentralized voting: a self-tallying voting system using a smart contract on the ethereum blockchain. In: International Conference on Web Information Systems Engineering. Springer; 2018, p. 18–35.

[75] Yang X, Yi X, Nepal S, Kelarev A, Han F. Blockchain voting: Publicly verifiable online voting protocol without trusted tallying authorities. Future Gener Comput Syst 2020.

[76] Yu B, Liu JK, Sakzad A, Nepal S, Steinfeld R, Rimba P, Au MH. Platform-independent secure blockchain-based voting system. In: International Conference on Information Security. Springer; 2018, p. 369–86.